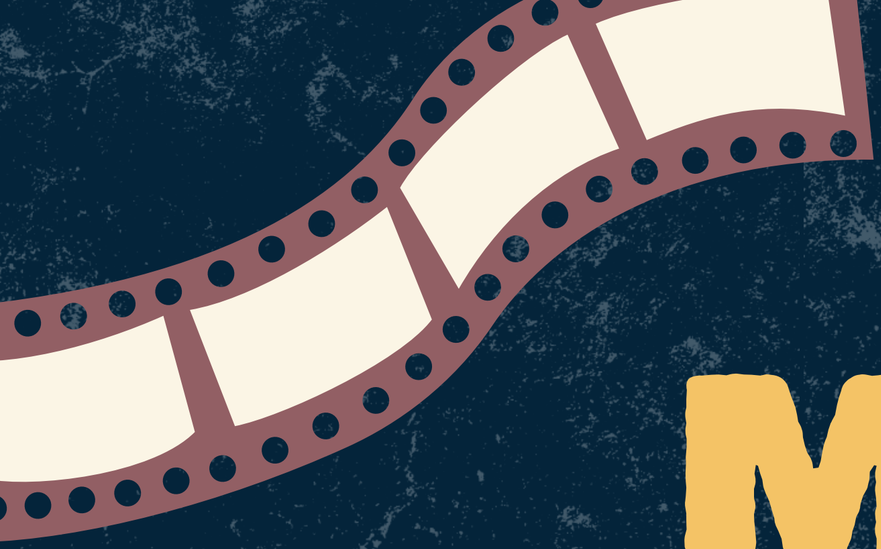
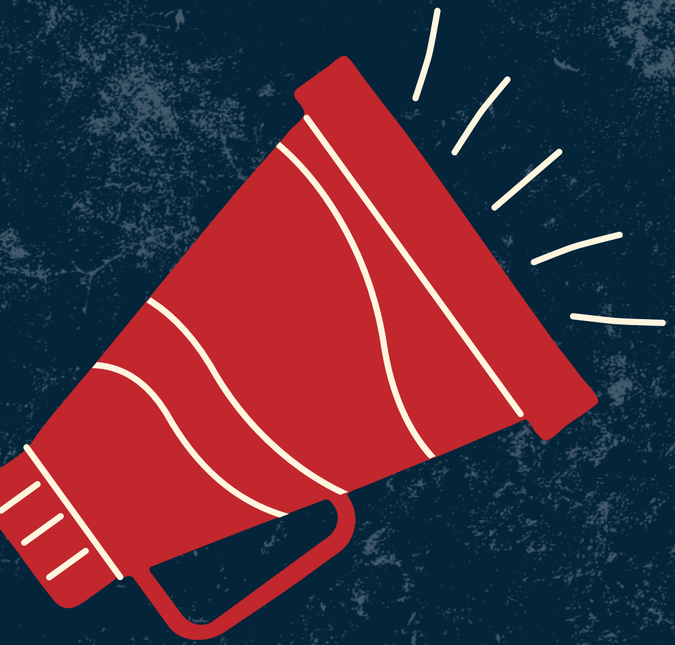


M-LANG

A Cinematic Approach to Coding

Team Members:

Ashutosh Kumbhar
Rajesh Sawant
Girish Nalawade
Mitesh Parab



WHAT iS M-LANG?

M-Lang is a custom-designed programming language that merges cinematic storytelling with software logic, offering a fresh and creative way to write code.

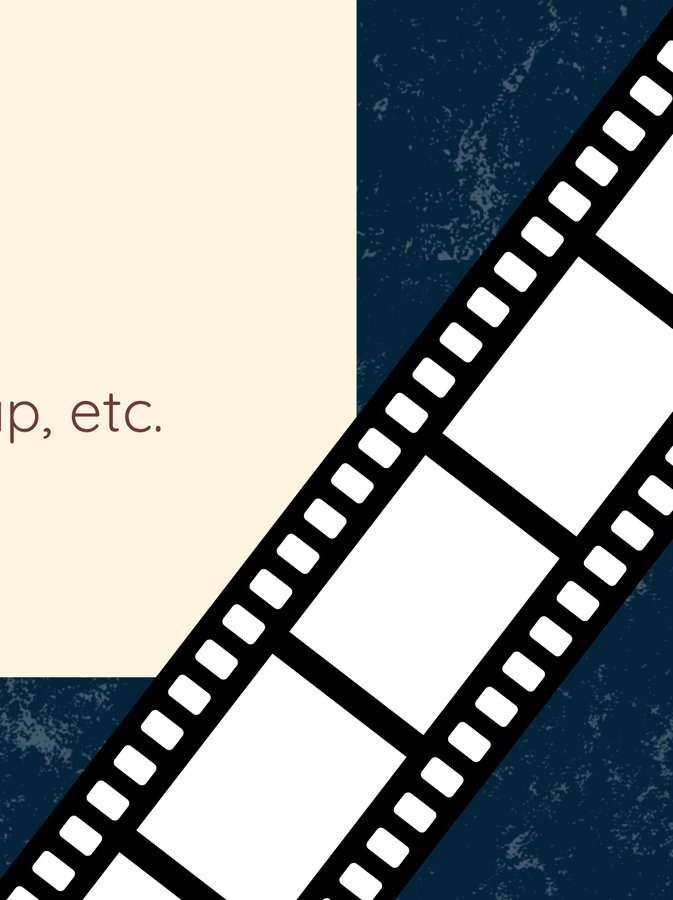
Inspired by screenplay writing, M-Lang transforms traditional programming constructs into a format that reads like a script, making code feel more narrative and expressive.

Key Characteristics:

- Narrative Syntax: Write logic in a way that feels like telling a story.
- Simple Yet Powerful: Supports core programming concepts like variables, expressions, printing, and control flow.
- Educational & Engaging: Ideal for beginners, creative coders, and educators who want to teach programming through storytelling.

LANGUAGE OVERVIEW

- Paradigm : Imperative
- Typing: Dynamic
- Execution: Interpreted
- Syntax Inspired by film editing: cast, cutIf, montage, wrap, etc.



KEY FEATURES

🎬 CINEMATIC STORYTELLING SYNTAX

Uses natural, screenplay-inspired keywords like cutIf, plotTwist, scene, and wrap, making code read like a movie script.

⚙️ INTERPRETED EXECUTION WITH ANTLR & PYTHON

M-Lang code is parsed using ANTLR and executed with a custom Python interpreter, enabling dynamic execution and flexibility.

🔁 FULL CONTROL FLOW SUPPORT

Supports if, else, while, for, and break/continue equivalents using cinematic metaphors like rollWhile, montage, pause, and replay.

📊 DYNAMIC TYPING AND EXPRESSION HANDLING

No need for static type declarations; supports arithmetic, booleans, ternary operations, and functions with inferred types.

TOOLS & TECHNOLOGIES

- **Python** – for building the interpreter and runtime environment
- **ANTLR4** – for defining and generating the lexer and parser from the grammar
- **VS Code** – the primary IDE used for grammar design and interpreter development
- **Graphviz + Pydot** – for generating parse tree visualizations
- **GitHub** – for version control and collaborative development
- **Slack** – for coordination and communication among team members

LANGUAGE DESIGN GOALS

NARRATIVE READABILITY AND THEMATIC EXPRESSION

M-Lang was designed to make programming feel like storytelling. By replacing conventional keywords with cinematic metaphors (e.g., cutIf instead of if, scene for function definitions), the language aims to improve readability and engagement. This approach makes the code intuitive and expressive, particularly for beginners, educators, or creative developers who appreciate narrative-driven logic.

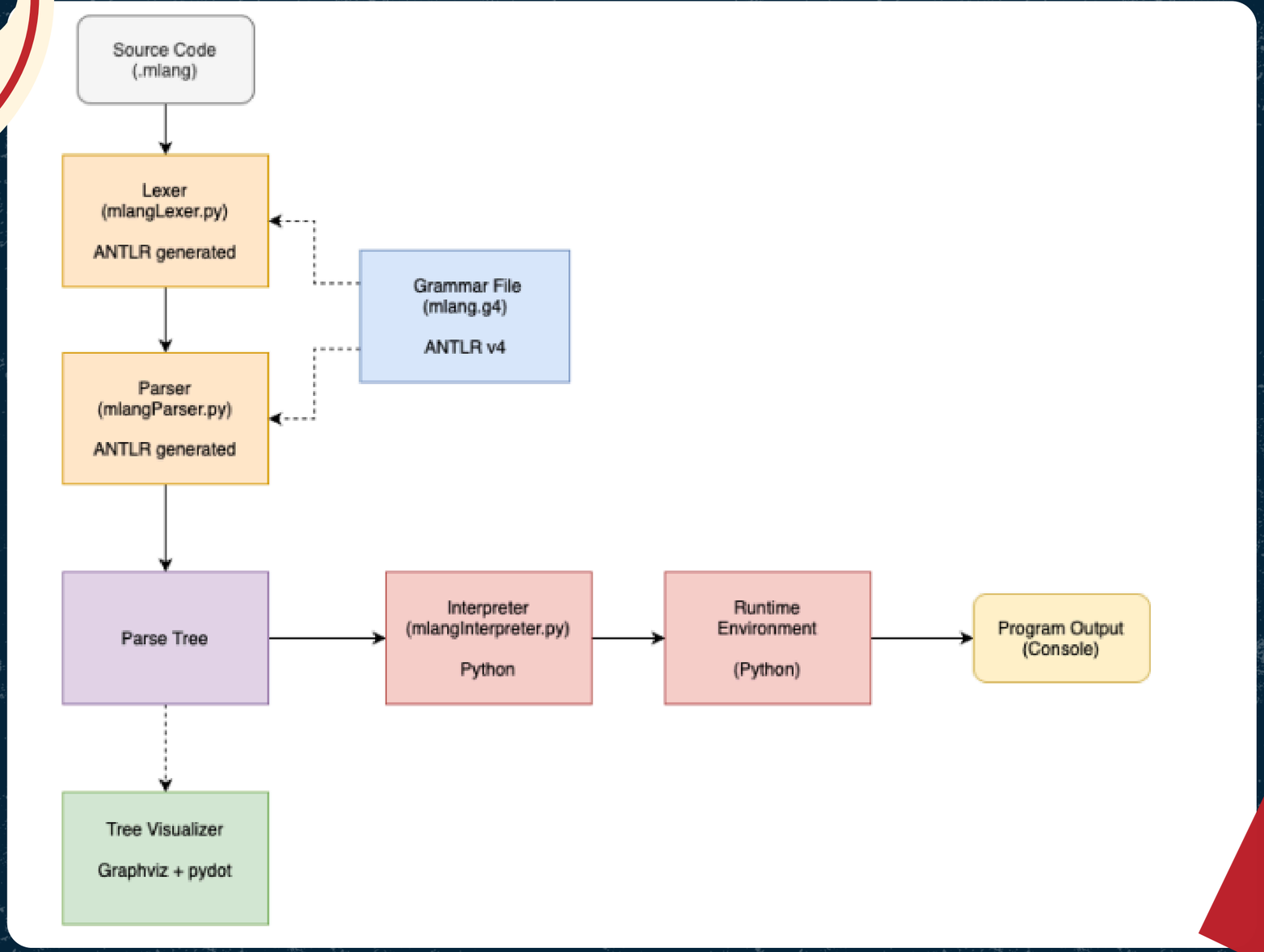
MODULAR AND SCALABLE GRAMMAR DESIGN

The grammar of M-Lang is modularly defined using ANTLR v4, allowing for easy extensions and refinements. From simple expressions to complex control structures and function calls, the design supports progressive enhancements without breaking existing syntax. This modularity not only simplifies maintenance but also encourages experimentation with new constructs like lambdas or object-oriented features in the future.

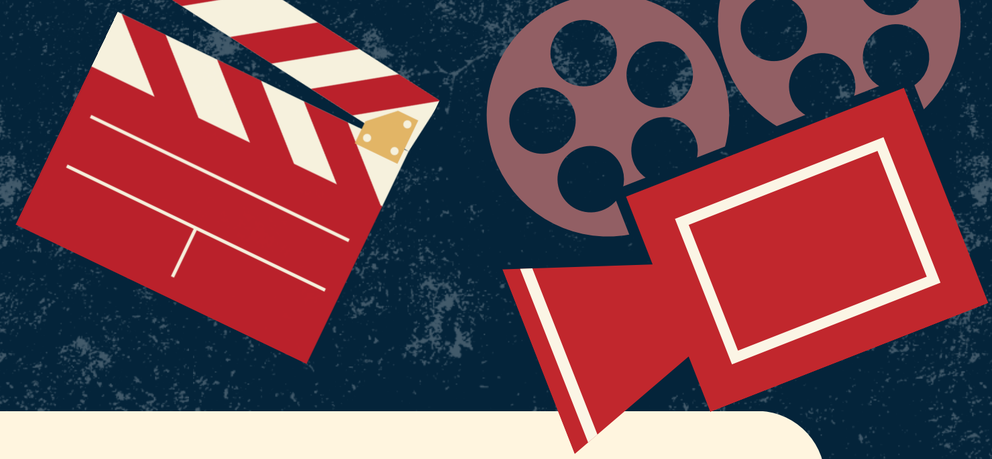
INTERPRETABILITY AND EXECUTION FLEXIBILITY

M-Lang follows an interpreted execution model, leveraging a tree-walking Python interpreter. This allows for immediate execution, better error reporting, and interactive debugging. Dynamic typing was chosen to provide more flexibility, allowing variables and functions to adapt based on usage, and making the language more forgiving and fluid for learners and rapid prototyping.

ARCHITECTURE



CORE LANGUAGE CONSTRUCTS iN M-LANG



DATA TYPES

- Primitive Types: INT, BOOL, STRING
- Type Inference: cast x is 10; // x inferred as INT

CONTROL FLOW

- Conditional: cutIf, altCut, plotTwist
- Loops:
- rollWhile condition action ... cut
- montage cast i is 0; i smallerThan 10; i is i + 1 action ... cut

FUNCTiONS

- Defined using scene ... action ... cut
- Supports parameters (with) and return values using wrap
- Example:

```
scene square with x
action
    wrap x * x;
cut
```

SYNTAX OVERVIEW — STATEMENTS & CONTROL FLOW

MLang Syntax	Meaning	Traditional Equivalent
cast x is 5;	Variable declaration	let x = 5
x is 10;	Assignment	x = 10
say x;	Print statement	print(x)
cutIf cond action ... cut	if statement block	if cond: ...
altCut cond action ... cut	else if block	elif cond: ...
plotTwist action ... cut	else block	else: ...
montage cast i is 0; i < 5; i is i + 1 action ... cut	for loop	for i in range(0, 5): ...
rollWhile cond action ... cut	while loop	while cond: ...
pause;	break	break
replay;	continue	continue

SYNTAX OVERVIEW — EXPRESSIONS & FUNCTIONS

MLang Syntax	Meaning	Traditional Equivalent
scene f with x action ... cut	Function definition	def f(x): ...
call f with 5;	Function call	f(5)
wrap x;	Return value	return x
a cut? b plotTwist c	Ternary conditional	b ? a : c
a + b, a - b, a * b	Arithmetic operators	a + b, a - b, a * b
not x	Logical NOT	not x
a andAlso b / a orElse b	Logical AND / OR	a and b / a or b
a sameAs b / a notSame b	Equality comparison	a == b / a != b
a smallerThan b / a biggerOrEqual b	Comparison operators	a < b / a >= b
“Hi “ + name	String concatenation	“Hi “ + name

RESERVED KEYWORDS

Category	Reserved Keywords
Language Syntax	cast, is, say, note:, action, cut
Control Flow	cutIf, altCut, plotTwist, rollWhile, montage, pause, replay, cut?
Functionality	scene, with, wrap, call
Boolean Literals	truth, lie
Comparison Ops	sameAs, notSame, biggerThan, smallerThan
Logic Operators	andAlso, orElse, not

SAMPLE CODE

Python:

```python
pill = "red"

if pill == "red":
    print("You chose to know the truth.")
elif pill == "blue":
    print("You chose comfort and illusion.")
else:
    print("Indecision is also a choice.")
```

This Python snippet uses an if-elif-else structure to simulate a choice between pills, printing a message based on the selected value.

M-Lang

```
note: The Oracle presents the choice

cast pill is "red";

cutIf pill sameAs "red"
action
  say "You chose to know the truth.";
cut

altCut pill sameAs "blue"
action
  say "You chose comfort and illusion.";
cut

plotTwist
action
  say "Indecision is also a choice.";
cut
```

M-Lang expresses the same logic using cinematic keywords like cutIf, altCut, and plotTwist, turning the conditional structure into a readable, screenplay-style flow.

BOSS FiGHT SiMULATiON PROGRAM iN MLANG

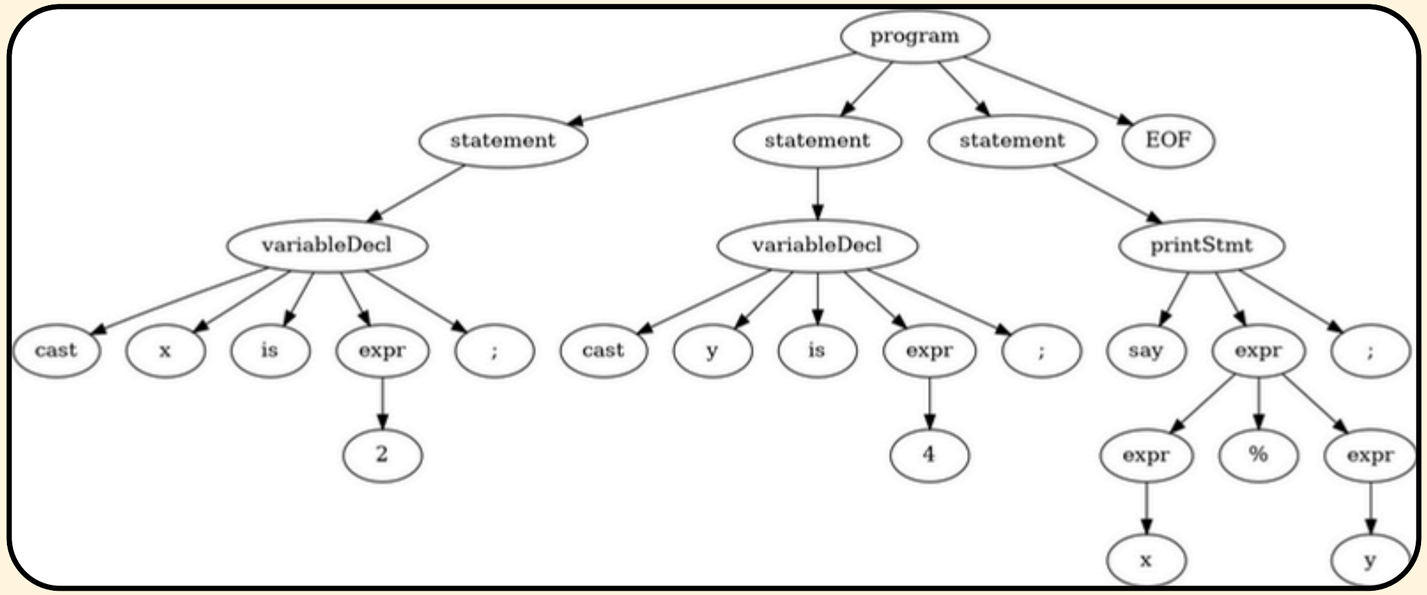
bossFight.mlang

```
cast enemyHealth is 100;
cast attackPower is 40;

say "Boss Health:" + enemyHealth;

rollWhile enemyHealth biggerThan 0 action
  say "Attacking with power: " + attackPower;
  enemyHealth is enemyHealth - attackPower;

  cast status is enemyHealth smallerOrEqual 0 cut? "Victory" plotTwist "Still fighting...";
  say status;
cut
```

This cinematic program simulates a battle loop where a hero repeatedly attacks a boss until the enemy's health drops to zero. It showcases:

- Variable declarations (cast)
- Loops (rollWhile)
- Arithmetic operations and assignments
- Ternary conditional logic (cut? ... plotTwist ...)
- String concatenation and dynamic printing

SAMPLE PARSE TREE



Code:

```
cast x is 2;
cast y is 4;
say x % y;
```

FUTURE SCOPE

🔧 LANGUAGE EXTENSIONS

Add support for user-defined classes, objects, and exception handling to broaden application domains.

🌐 WEB-BASED IDE

Develop an interactive web playground with syntax highlighting and real-time execution for learning and experimentation.

📦 MODULAR PROGRAMMING

Introduce support for modules, imports, and namespaces to enable larger, more maintainable M-Lang programs.

📈 PERFORMANCE OPTIMIZATION

The camera is placed overhead or directly above the subject. Characters and objects are made to look small compared to their surroundings.

🤝 INTEGRATION WITH OTHER TOOLS

The camera is positioned directly beneath the subject. It is often coupled with point-of-view shots when the character is looking up at something.

📚 EDUCATIONAL ADOPTION

Package M-Lang as a tool for teaching programming and logical thinking through storytelling in schools and creative courses.

THANK YOU